The International Journal Of Engineering And Science (IJES)
|| Volume || 3 || Issue || 5 || Pages || 36-44 || 2014 ||
ISSN (e): 2319 – 1813 ISSN (p): 2319 – 1805

Design and Implementation of Data Integrity System Using MD5 Cryptographic Hash Function for Federal Polytechnic Bauchi

[1]Esiefarienrhe Michael Bukohwo , [2]Fa'iz Ibrahim Jibia
[1]Maths/Computer Science Dept., University of Agric., Makurdi. Email: esiefabukohwo@gmail.com
[2]Computer Science Dept., Federal Polytechnic, Bauchi.

--ABSTRACT--
This paper seeks to present a data integrity checking system designed based on MD5 algorithm to ensure the integrity of files sent and received via transmission lines between departments and the Exams and record unit of the Federal Polytechnic, Bauchi. The data for the system were collected using observation, interviews and review of existing documentation. The Structured Systems Analysis and Design (SSADM) and Knowledge Engineering Methodologies were used in the problem analysis and design of the system. The program for the system was written using Microsoft Visual Basic 6.0 as part of Microsoft Studio. The system implementation shows that documents management greatly improved as document within the units were all encrypted making it impossible for falsification, results manipulations, theft and misuse.

KEYWORDS: *Md5, Cryptography, Hash Function, Data Integrity, Collision, Algorithm, Digital Signature*

Date of Submission: 26 February 2014 Date of Publication: 25 May 2014

I. INTRODUCTION

Every institution whether private or public will strive to protect the integrity of the information it generate, safeguard its transmission over a network as well as ensure it proper accessibility when the need arises. Safeguarding information is vital to the success of any business as both origin, use and store of the information is vital to its reliability. To ensure easy of data integrity checking without spending much, a cryptographic has function is therefore developed for use.Cryptographic hash function is a deterministic procedure that takes an arbitrary block of data and returns a fixed-size bit string, the (cryptographic) hash value, such that an accidental or intentional change to the data will change the hash value. The data to be encoded is often called the "message," and the hash value is sometimes called the message digests or simply digests

This Message Digest 5 (**MD5)** algorithm is a widely used cryptographic hash function that produces a 128-bit (16-byte) hash value and has been employed in a wide variety of security applications including checking data integrity. MD5 is one in a series of message digest algorithms designed by Professor Ronald Rivest of MIT [8]. When analytic work indicated that MD5's predecessor MD4 was likely to be insecure, MD5 was designed in 1991 to be a secure replacement.The MD5 algorithm is intended for digital signature applications, where a large file must be "compressed" in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem. The MD5 algorithm is designed to be quite fast on 32-bit machines. In addition, the MD5 Algorithm does not require any large substitution. It has since been found that MD5 is not collision resistant [3][4][5][6][7][8] and therefore not suitable for applications requiring SSL certificates, digital signatures, encryption of large and extremely sensitive document files. For extremely sensitive document, SHA-2 family of hash function may be recommended. The researchers used MD5 for the implementation of the security of documents of the exams and records units for the following reasons:

[1] The system is meant for internal use by staff within the polytechnic for the encryption of students records and students' information related. It is not meant for securing financial transactions and accounting functions of the polytechnic.

[2] The unique benefit of information technology is the ability to apply it to providing solutions to internal problem and the ability to use local resources e.g. Manpower, limited funds etc to bring about positive change in a system. To this regard, this system has been able to introduce such changes by securing its own records and using its own personnel to design and implement the security system.

[3] The system so design is upgradable to higher platform as the module that implement the MD5 function can easily be replaced by any other function e.g. SHA-2. This is the advantage of modular programming utilized by the researcher.

Background of the organisation

The Federal Polytechnic Bauchi is one of the seven Polytechnics established by the Military Government of Nigeria under Decree No. 33 of July 1979. The Examinations and record unit is one of the units of the Academic Registry of the Polytechnic and is headed by an Academic Officer who reports directly to the Registrar. It handles all Examination materials, student's records and all issues pertaining to students' academic performance. Also, the unit is in charge of holding student secret file which are generated by the admission office of the Polytechnic during registration and later forwarded to the examinations and record unit. Control cards, registration forms as well as acceptance letter from students are compiled by the exams and record unit and later forwarded to the various academic departments of the institution. At the end of each academic session, the unit prepares and issue result to the students (both the semester and the final result) as well as the academic transcript.It is not surprising therefore that this unit has been plaque with problems related to results falsification, grade adjustment, missing students' records, document forgery etc. It is the frequency of the occurrence of the above problems that prompted the design this data integrity checking system for the Examinations and record unit of the polytechnic with the goal of ensuring the integrity of transmitted document within departments and units of the institution.

II. MATERIALS AND METHODS

The methodology used in this research is the Structured System Analysis and Design (SSADM) which is an accepted Software Engineering Methodology for software. Also used is the Expert System methodology, which involved knowledge engineering process of inference and knowledge-based [1]. SSADM involves a thorough study of all sets of interacting entities, including computer systems analysis. This methodology is closely related to requirements engineering or operations research. It is an explicit formal inquiry carried out to by the decision maker to identify a better course of action and to make a better decision about a given system. The development of any computer-based information system includes a systems analysis phase which produces or enhances the data model which itself is a precursor to creating or enhancing a data repository. There are a number of models utilized in system analysis. In the design of the research, we utilized the Spiral model [14] because of its ability to incorporate risk management in its framework and it entails. The use of this model enable enables:

- the development of a feasibility study, involving determining whether a project is economically, socially, technologically and organizationally feasible.
- conducting fact-finding measures, designed to ascertain the requirements of the system's end-users. These typically span interviews, questionnaires, or visual observations of work on the existing system.
- simulating how the end-users would operate the system (in terms of general experience in using computer hardware or software), what the system would be used for etc.

In carrying out a feasibility study of the system, two data collection techniques were used: Observation and interview. All documents that are handled by the unit were made available which includes students' files, course registration details, course assessment and examination details, file naming convention and communication procedures within and between departments and units. Also observed were the unit staff at work so as to gather information about document handling and possible problems experienced in daily performance of task of securing document. Interviews were also conducted for the staff to elicit facts about the system. At the end of this process, the following were obvious which contributed immensely to the development of the system:

- The volume of transaction performed on a daily basis by the unit
- Problems encountered in the application of the manual system of document handling and safety.
- The amount of storage space that will likely be idea for the unit
- Communication needs among various departments and units within the polytechnic and how these needs could be adequately satisfied

System design entails the construction of a model in a logical manner that will meet the requirement of the system.

The Data Integrity System

The system that was developed comprises of two separate modules: File hashing module called checks, and hash verification module (Fig. 1).

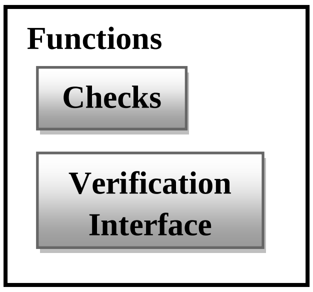


Fig. 1: The system Modules

File Hashing Module: The entire system is intended at checking/verifying the integrity of files that are transferred over the network from any of the polytechnic departments to the examinations and record unit. This module consists of set of codes that will accomplish the above stated functionality.

Hash Verification Module: Verification means to check and confirm that the files that were sent are the actual files received without any change or modification within or along the network [2]. This is accomplished by hashing the copy of the received files and a comparison is made between the newly generated hash and the original hash which was generated by the sender before the files was sent over the network. If the two hashes are found to be the same, conclusion is reached that the integrity of the files has not been tempered with during transmission. The file hash verification module is the one that accomplish the above stated functionality.

We present the algorithm that achieves both the file handling and hash verification procedures:

MD5 Algorithm Description

We begin by supposing that we have a b-bit message as input, and that we wish to find its message digest. Here b is an arbitrary nonnegative integer; b may be zero, it need not be a multiple of eight, and it may be arbitrarily large. We imagine the bits of the message written down as follows:

m_0 m_1 ... m_{b-1}

The following five steps are performed to compute the message digest of the message [13].

Step 1. Append Padding Bits

The message is "padded" (extended) so that its length (in bits) is congruent to 448, modulo 512. That is, the message is extended so that it is just 64 bits shy of being a multiple of 512 bits long.

Padding is always performed, even if the length of the message is already congruent to 448, modulo 512. Padding is performed as follows: a single "1" bit is appended to the message, and then "0" bits are appended so that the length in bits of the padded message becomes congruent to 448, modulo 512. In all, at least one bit and at most 512 bits are appended.

Step 2. Append Length

A 64-bit representation of b (the length of the message before the padding bits were added) is appended to the result of the previous step. In the unlikely event that b is greater than 2^64, then only the low-order 64 bits of b are used. (These bits are appended as two 32-bit words and appended low-order word first in accordance with the previous conventions.)

At this point the resulting message (after padding with bits and with b) has a length that is an exact multiple of 512 bits. Equivalently, this message has a length that is an exact multiple of 16 (32-bit) words. Let M[0 ... N-1] denote the words of the resulting message, where N is a multiple of 16 [10][11].

Step 3. Initialize MD Buffer

A four-word buffer (A,B,C,D) is used to compute the message digest.

Here each of A, B, C, D is a 32-bit register. These registers are initialized to the following values in hexadecimal, low-order bytes first):

word A: 01 23 45 67
word B: 89 ab cd ef
word C: fe dc ba 98
word D: 76 54 32 10

Step 4. Process Message in 16-Word Blocks

We first define four auxiliary functions that each take as input three 32-bit words and produce as output one 32-bit word.
F(X,Y,Z) = XY v not(X) Z
G(X,Y,Z) = XZ v Y not(Z)
H(X,Y,Z) = X xor Y xor Z
I(X,Y,Z) = Y xor (X v not(Z))
In each bit position F acts as a conditional: if X then Y else Z.
The function F could have been defined using + instead of v since XY and not(X) Z will never have 1's in the same bit position.) It is interesting to note that if the bits of X, Y, and Z are independent and unbiased, the each bit of F(X,Y,Z) will be independent and unbiased.
The functions G, H, and I are similar to the function F, in that they act in "bitwise parallel" to produce their output from the bits of X, Y, and Z, in such a manner that if the corresponding bits of X, Y, and Z are independent and unbiased, then each bit of G(X,Y,Z),
H(X,Y,Z), and I(X,Y,Z) will be independent and unbiased. Note that the function H is the bit-wise "xor" or "parity" function of its inputs.
This step uses a 64-element table T[1 ... 64 [9][12]. constructed from the sine function. Let T[i] denote the i-th element of the table, which is equal to the integer part of 4294967296 times abs(sin(i)), where I is in radians. The elements of the table are given in the appendix.
Do the following:

```
 /* Process each 16-word block. */
For i = 0 to N/16-1 do
/* Copy block i into X. */
For j = 0 to 15 do
Set X[j] to M[i*16+j].
end /* of loop on j */
 /* Save A as AA, B as BB, C as CC, and D as DD.     AA = A
BB = B
CC = C
DD = D
/* Round 1. */
/* Let [abcd k s i] denote the operation
a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD  0  7  1]  [DABC  1 12  2]  [CDAB  2 17  3]  [BCDA  3 22  4]
 [ABCD  4  7  5]  [DABC  5 12  6]  [CDAB  6 17  7]  [BCDA  7 22  8]
[ABCD  8  7  9]  [DABC  9 12 10]  [CDAB 10 17 11]  [BCDA 11 22 12]
[ABCD 12  7 13]  [DABC 13 12 14]  [CDAB 14 17 15]  [BCDA 15 22 16]
/* Round 2. */
/* Let [abcd k s i] denote the operation
a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD  1  5 17]  [DABC  6  9 18]  [CDAB 11 14 19]  [BCDA  0 20 20]
 [ABCD  5  5 21]  [DABC 10  9 22]  [CDAB 15 14 23]  [BCDA  4 20 24]
[ABCD  9  5 25]  [DABC 14  9 26]  [CDAB  3 14 27]  [BCDA  8 20 28]
 [ABCD 13  5 29]  [DABC  2  9 30]  [CDAB  7 14 31]  [BCDA 12 20 32]
/* Round 3. */
/* Let [abcd k s t] denote the operation
a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD  5  4 33]  [DABC  8 11 34]  [CDAB 11 16 35]  [BCDA 14 23 36]
[ABCD  1  4 37]  [DABC  4 11 38]  [CDAB  7 16 39]  [BCDA 10 23 40]
[ABCD 13  4 41]  [DABC  0 11 42]  [CDAB  3 16 43]  [BCDA  6 23 44]
[ABCD  9  4 45]  [DABC 12 11 46]  [CDAB 15 16 47]  [BCDA  2 23 48]
/* Round 4. */
/* Let [abcd k s t] denote the operation
a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s). */
 /* Do the following 16 operations. */
 [ABCD  0  6 49]  [DABC  7 10 50]  [CDAB 14 15 51]  [BCDA  5 21 52]
 [ABCD 12  6 53]  [DABC  3 10 54]  [CDAB 10 15 55]  [BCDA  1 21 56]
 [ABCD  8  6 57]  [DABC 15 10 58]  [CDAB  6 15 59]  [BCDA 13 21 60]
 [ABCD  4  6 61]  [DABC 11 10 62]  [CDAB  2 15 63]  [BCDA  9 21 64]
/* Then perform the following additions. (That is increment each of the four registers by the value it had before
this block was started.) */
A = A + AA
B = B + BB
C = C + CC
D = D + DD end /* of loop on i */
```

Step 5. Output

The message digest produced as output A, B, C, D. That is, we begin with the low-order byte of A, and end with the high-order byte of D.

System Block Diagram: This depicts the general structure of the new system (Fig. 2).

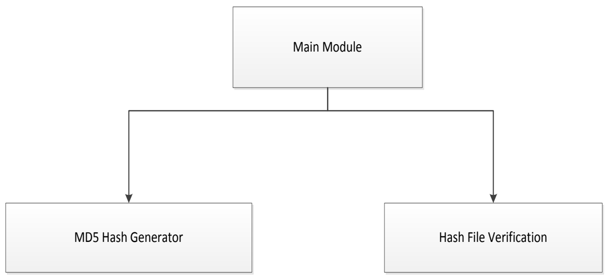


Fig. 2: System Architecture:

Data Flow Strategy: This gives a verbal explanation of the sequence of operations that must be executed in order to achieve the aim of the system. The system is composed of two modules. When the system is implemented, the two modules are displayed, numbered 1 & 2; from which the user selects the one to operate. If the user selects 1, the hash generation module is activated otherwise the hash verification module is activated (Fig. 3).

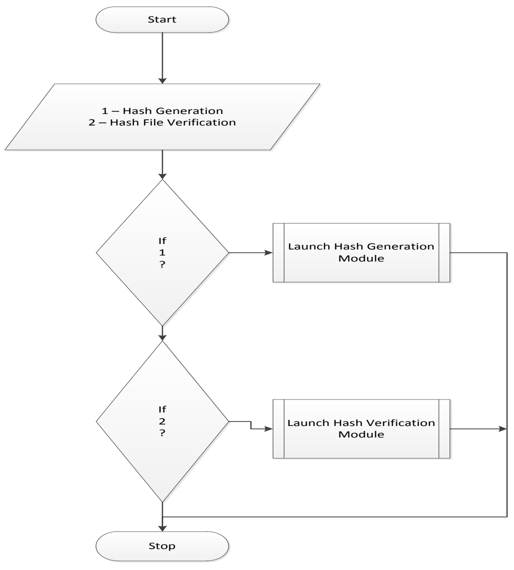

Start
1 – Hash Generation
2 – Hash File Verification
If
1
?
Launch Hash Generation Module
If
2
?
Launch Hash Verification Module
Stop

Fig. 3: Data Flow Strategy

After launching the hash generation module interface, the user then selects the files to hash, calculation is perform based on the MD5 algorithm which take five steps (Fig. 4) to compute the hashes, the steps includes Padding, Appending the length of the message, Initializing the MD buffer, Processing the message in 16 word blocks and finally outputting the hashes, the hashes are now save to the MD5 files.

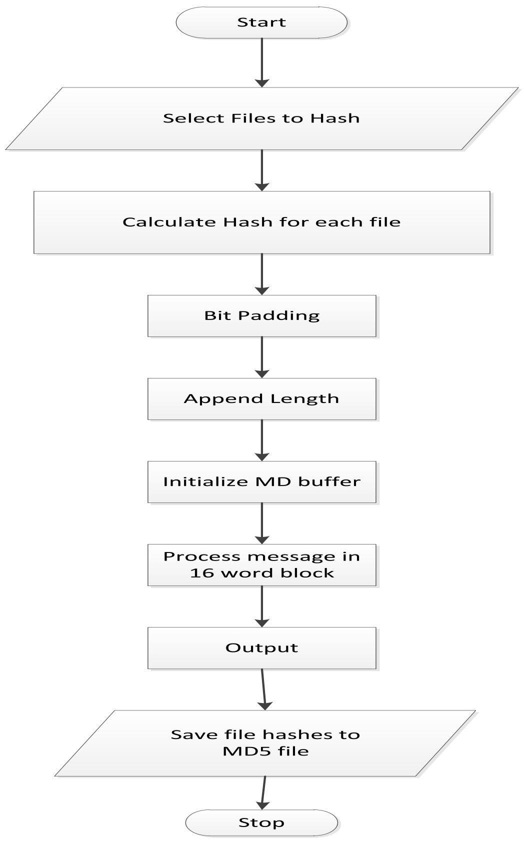

Start
Select Files to Hash
Calculate Hash for each file
Bit Padding
Append Length
Initialize MD buffer
Process message in 16 word block
Output
Save file hashes to MD5 file
Stop

Fig. 4: Flowchart of Hash Generation Module

After activating the Hash Verification Module interface, the user can now input/read the hash files, load the files and their corresponding hashes and also select the location of the files to verify. The new hashes are then calculated and compared against the original hashes, if the hashes are found to be the same, a report is displayed that "Verification Successful" or else "Verification error" is displayed (Fig. 5).

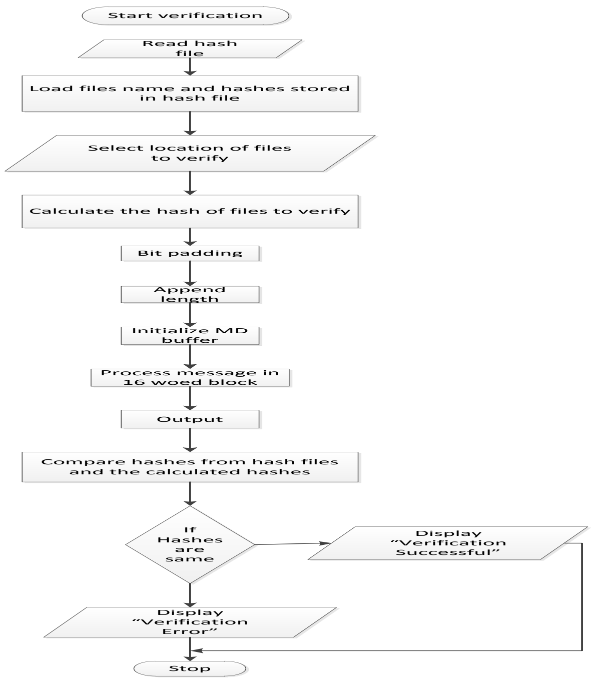

Start verification
Read hash file
Load files name and hashes stored in hash file
Select location of files to verify
Calculate the hash of files to verify
Bit padding
Append length
Initialize MD buffer
Process message in 16 woed block
Output
Compare hashes from hash files and the calculated hashes
If Hashes are same
Display "Verification Successful"
Display "Verification Error"
Stop

Fig. 5: Flowchart of Hash Verification Module

System Input Form Design and Sample Input

This is the dialog box that enables users to encode their files and thus specify how data are entered into the system. Fig. 6a shows the input dialog form.

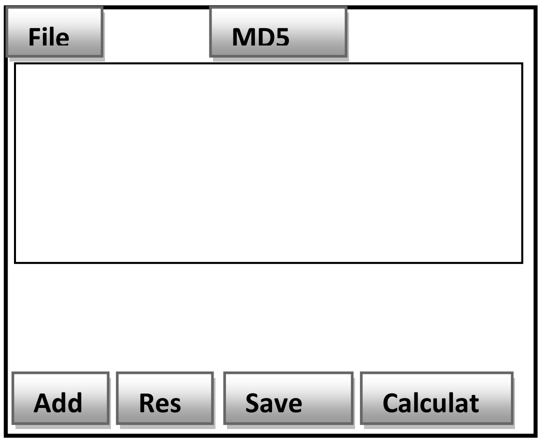


Fig 6a: Input Form

The input process entails the user to browse for the file to be hash and to also verify if it is the actual file selected. The user simply click "Browse for the Hash file" and then click "verify" as shown in Fig. 6b.

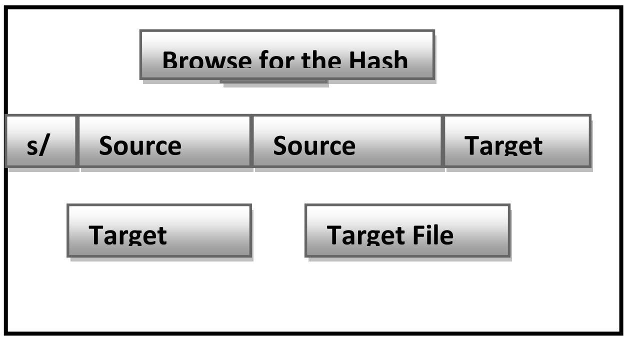
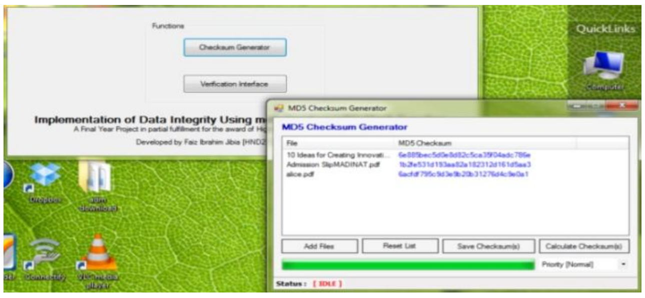


Fig. 6b: Sample System Input

The System Output Design and Sample Form

Similarly, another interface is design to display the result of the verification exercise, which comprises of the name of the source file, source/initial hashes, targeted/new hashes and status of the verification (Fig, 7a). The output generated by the system based on the input of Fig. 6b is shown in Fig. 7b.



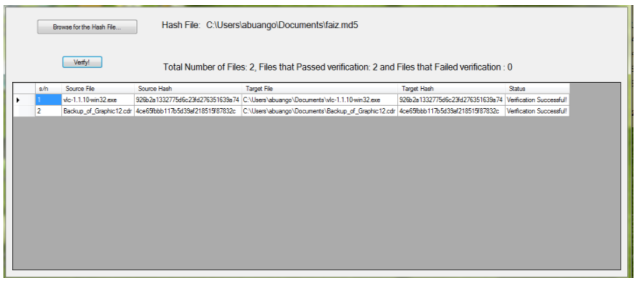
Implementation of Data Integrity Using m
A Final Year Project in partial fulfillment for the award of H
Developed by Faiz Ibrahim Jibia [HND2

MD5 Checksum Generator

File	MD5 Checksum
10 Ideas for Creating Innovati...	6e885bec5d0e8d82c5ca35f04adc786e
Admission SlipMADINAT.pdf	1b2fe531d193aa82a182312d161d5aa3
alice.pdf	6acfdf795c9d3e9b20b31276d4c9e0a1

Status : [IDLE]

Fig 7a: Output Form



Hash File: C:\Users\abuango\Documents\faiz.md5

Total Number of Files: 2, Files that Passed verification: 2 and Files that Failed verification : 0

s/n	Source File	Source Hash	Target File	Target Hash	Status
1	vlc-1.1.10-win32.exe	926b2a1332775d6c23fd276351639a74	C:\Users\abuango\Documents\vlc-1.1.10-win32.exe	926b2a1332775d6c23fd276351639a74	Verification Successful!
2	Backup_of_Graphic12.cdr	4ce65fbbb117b5d39af218515f87832c	C:\Users\abuango\Documents\Backup_of_Graphic12.cdr	4ce65fbbb117b5d39af218515f87832c	Verification Successful!

Fig. 7b: Sample System Output

Development tools Utilized

The data integrity checking system was developed using several technologies which are technically sufficient to build a reliable and well-maintained platform. Below are the tools that were utilized:

- **Microsoft visual studio2010:** is an integrated development environment (IDE) from Microsoft. It is used to develop console and graphical user interface applications along with Windows Forms applications, web sites, web applications, and
- web services in both native code together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework and Microsoft Silver light.

System Requirements

The major requirements for the successful implementation and operation of the system are listed below:

- Pentium 4 processor or higher
- 1 gb RAM or higher
- .Net Framework 3.5 or higher
- USB 2.0 port or higher

User Operational Manual for *the Data integrity checking system* guides the user on how he/she will go about using the Software.

Details

Once the software is launched from desktop

The Main/Welcome Screen appears on screen.

The Graphical User Interface (GUI) of the main screen has the following:

[1] Checksum Generator Button – when clicked, it enables the user to insert the file(s) that will be hashed

[2] Verification interface Button – On clicking this button, the verification interface is open, where two button are made available for the user, the browse button and verify button. on clicking the browse button, a window is open for the user to choose the desired hash file for comparison. After selecting the hash files, verify button is clicked which execute the comparison and display the result.

[3] Exit Button – This button exit the interface once clicked.

SUMMARY

Ensuring the integrity of files transferred from one department/unit to the other has been a major concern in most organizations, institutions, businesses and industries today. Hard copies of documents are carried manually from one department to another within the same organisation. Even with the advent of electronic mails, security of document is left in the hands of the owner of such document. In most situation, the security provided by the network architecture in cases of electronic mails may not be enough to ensure a secured transmission of document. It is even worst when the medium of manual. MD5 is one among various algorithms that can be used to implement a system that will ensure the integrity of document being transferred over a connection from one department/unit to the other.This research therefore focuses on the design of software that utilises the concept of the MD5 algorithm to overcome the problems of document management of the above institution of higher learning. The system is composed of two modules with two different functionalities. The first module is called the Hash Generation module, which is responsible for hashing a file (generate the checksums/hashes of the file) before sending it. The second module, which is called the Verification Module, receives the encrypted file, performs the verification function; and make comparison between the source file and the target file and a status report is generated at the end of the comparison, which determine whether what was actually sent has been received without any alteration or modification. In no small way, the system has actually be helpful to the institution since its implementation.

III. CONCLUSION

Conclusively, the developed system would solve the associated problems with the traditional/manual system. Delay and time consumption will surely be eliminated with the documents being sent over a secure network connection, and an effective & efficient integrity checking system that will guaranty the authenticity of the received document.Similarly, with the successful implementation of this system, confidentiality of messages will definitely be maintained as only authorised personnel of an organization can have access to their systems, unlike the manual system in which junior staffs are charged with the responsibility of delivering mails in the physical sense, so this makes it more prone for the message to be exposed to an outsider who does not have the authority of seen such a confidential message. The prototype of this system has been tasted with different forms of data and it is achieved that the system successfully generates hashes/checksums of a given data, stores them and also performs the verification test and finally give a status report about the message

REFERENCES:

[1] Anigbogu S. O And Inyiama H. C. (2006); Artificial Intelligence-Based Medical Diagnostic Expert System For Malaria And The Related Ailments, Journal Of Computer Science And Its Applications, Vol 12, No 1 P. 3-8.

[2] Arjen Lenstra Et Al, (2005) Colliding X.509 Certificates, Cryptology Eprint Archive Report. Http://En.Wikipedia.Org/Wiki/MD5

[3] (11 Aug, 2011.)

[4] Bert D. B And Antoon B. (1993). Collisions For The Compression Function Of MD5. Available Via: Http://En.Wikipedia.Org/Wiki/MD5 On 11 Aug, 2011.

[5] Bellare M. And Rogaway, P. (2005). Introduction To Modern Cryptography. Available Via: Http://En.Wikipedia.Org/Wiki/Cryptography On 14 Sept, 2011.

[6] Black F. L. (2003). All About The MD5 Algorithm And How Php Uses It To Encrypt Users' Passwords. Avaiable Via: Http://Www.Neothermic.Com/Kb/Article40.Html On 11 Aug, 2011.

[7] Boritz, J. E (2000) "IS Practitioners' Views On Core Concepts Of Information Integrity". International Journal Of Accounting Information Systems. Elsevier. Vol 2 No 1.

[8] Http://En.Wikipedia.Org/Wiki/Data_Integrity

[9] (12 Aug, 2011.)

[10] Cycom C. (2006) The History Of Cryptography. Available On: Http://En.Wikipedia.Org/Wiki/Cryptography On 14 Sept, 2011

[11] Dobbertin H. (1996). The Status Of MD5 After A Recent Attack. Ttp://En.Wikipedia.Org/Wiki/MD5

[12] (13 Aug, 2011)

[13] Garry C. K. (2011), An Overview Of Cryptography. Www.Garykessler.Net/Library/Crypto.Html On 14 Sept, 2011.

[14] Katz J. An Lindell Y. (2006) "Introduction To Modern Cryptography." Http://En.Wikipedia.Org/Wiki/Digital_Signature Retrieved On: (7th Aug, 2011)

[15] Sasaki Y. And Kazumaro A. (2009). Finding Preimages In Full MD5 Faster Than Exhaustive Search Springer Berlin Heidelberg. Http://En.Wikipedia.Org/Wiki/MD5 Retrieved On 7th Aug 2011

[16] Vlastimil K. (March 2005): Finding MD5 Collisions – A Toy For A Notebook, Cryptology Eprint Archive Report

[17] Http://En.Wikipedia.Org/Wiki/MD5 Retrieved On (10th Sept, 2011)

[18] Xiaoyun W. And Hongbo Y. (2009) "How To Break MD5 And Other Hash Functions"

[19] Http//En.Wikipedia.Org/Wiki/MD5 Retrieved On (10th Sept, 2011).

[20] Boehm, B. (1988); A Spiral Model Of

[21] Software Development And Enhancement,

[22] IEEE Computer Society 21(5): 61-72